基于大规模古文语料库的词典构建及分词技术研究

邢付贵[1,2]　朱廷劭[2*]

[1]（中国科学院大学 北京 10049）

[2]（中国科学院心理研究所，北京 100101）

摘要 ：古文献的研究有助于传统文化的继承与发扬，而古文分词则是利用自然语言处理技术对古文献进行分析的重要环节，但由于缺少规范的数据资料而没有像现代汉语分词取得突破性进展。当前互联网拥有大量古汉语文本和词典方面的数据资料，但是这些数据分散，没有得到有效地整合。本文提出采集互联网非结构化古汉语数据，经过数据清洗和预处理抽取出一个古汉语基础词典，然后再利用互信息、信息熵、位置成词概率相结合的新词发现方法从大规模古籍文本中抽取古汉语候补词典，最终将基础词典与候补词典融合，利用正向最大匹配实现对古文的分词。与开源的分词器甲言在基于词典的分词方面比较后 F 值提高了 14%，取得了良好的效果，结果证明本文构建的分词器可以应用在古汉语文本分词上。

关键词：古汉语分词；大数据；语料库；Apache Spark

分类号：B849

Segmentation Dictionary construction and research of word segmentation based on large-scale ancient Chinese Corpus

Fugui Xing[1,2] Tingshao Zhu[2*]

[1]（University of Chinese Academy of Sciences, Beijing 10049）

[2] (Institute of Psychology，Chinese Academy of Sciences Beijing 100101）

Abstract： The corpus The study of ancient documents is helpful for the inheritance and development of traditional culture, and ancient Chinese word segmentation is an important part of the analysis of ancient documents by using natural language processing technology, but due to the lack of standard data, it has not made a breakthrough like modern Chinese word segmentation. At present, there are a large number of ancient Chinese text and dictionary data on the Internet, but these data are scattered and not effectively integrated. This paper proposes to collect unstructured ancient Chinese data on the Internet and extract a basic dictionary of ancient Chinese through Big Data processing, then extract the candidate Dictionary of ancient Chinese from large-scale ancient Chinese texts by using the new word discovery method

combining mutual information, information entropy and position word probability. Finally, we integrated the basic dictionary and the addition dictionary and used the forward maximum matching to tokenize ancient Chinese text. Compared with the open-source ancient Chinese tokenizer "Jiayan", the F value which segment based on the dictionary was improved by 14%, and good results were achieved. The result showed that this tokenizer constructed in this paper can be applied to ancient Chinese text word segmentation.

Key words: ancient Chinese word segmentation; Big Data; corpus; Apache Spark

1 引言

中国古代文献是一个宝藏，从古汉语文本中挖掘有价值的信息在考古中有很重要的意义[1]。通过对古文献的分析研究，有助于进一步弘扬传统文化，而古文分词则是利用自然语言处理技术对古文献进行分析的重要环节。由于古文和现代汉语无论是在词汇还是在语法上都有很大的不同，探索古文的有效分词是亟待解决的问题。

相关学者在古文分词方面做了大量研究。严顺 [2] 提出了基于条件随机场(CRF)的古汉语分词模型；王晓玉等 [3] 将 CRFs 模型和词典相结合用于古汉语的分词；钱智勇等 [4] 使用隐马尔科夫模型对《楚辞》进行了自动分词标注的研究实验,取得了较好的效果；李筱瑜等 [5] 将互信息，信息熵相结合进行新词发现，并与词典信息结合用于古籍文本分词研究，但是准确度并不高。

本文利用大规模网络古文语料库，通过对海量古文资料的分析[6]，构建古文词典。首先，通过爬虫从互联网采集古汉语相关的数据集，经过数据清洗和转换得到一个古汉语基础词典，这个基础词典是由互联网的非结构化数据转换所得，囊括了大部分古汉语字词，但是并不能保证古汉语词汇的全面性，所以本文进一步采用互信息、信息熵[8]和位置成词概率[7]相结合的新词发现方法，从大规模古籍文本中抽取古汉语词汇并得到候补词典，以此来弥补基础词典的不足。在融合得到的新词典的基础上，本文利用正向最大匹配实现对古文的分词，并与开源古汉语分词器甲言[9]进行分词性能比较。

2 研究方法

2.1 基础词典的构建

在互联网发达的今天，互联网上有很多古汉语相关的数据，例如收录词典的网站，汉文学网[25]，词典网[26]，汉典[27]，在线汉语字典[28]，国学大师[29]等，Github 上也有很多开放的古汉语数据集，百科上也有古汉语相关的文本资料。

本文统一将数据抽取到 Hadoop,存储在 hdfs 上。Hive 用于数据的统计，在分布式数据索引方面选择 Apache Elasticsearch[11] 用于筛选古汉语数据集，Spark 用于数据处理。

总体流程如图 1 所示。

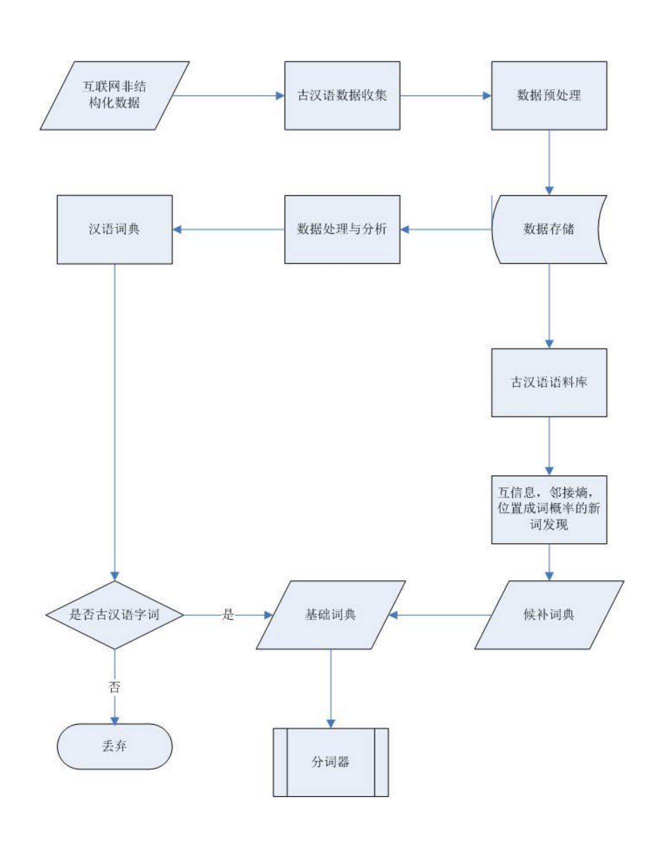


图 1 古文词典构建的处理流程

本文通过调查并选取了与古汉语词汇相关的网站，通过分析它们的 robots 协议和许可限制，最终确定可以用于爬虫采集数据的网站。经过数据的采集，转换，抽取了 22203 个字，364761 个词语。但是这些字和词语并不完全是古汉语，还包括一些现代字和词。这就需要对数据进行进一步处理，将现代字和词过滤掉。但是人工对这些字词进行筛选的工作量是巨大的，并且存在人为的误差，本文通过搜索古文语料库实现对古汉语字词的筛选。为了判断某一个字或者某一个词是否为古汉语字词，在古汉语语料库中进行查找，如果在语料库中存在，就被认定为古汉语字或者古汉语词语，否则不是。

Github 上有很多开源的古汉语语料库，其中，汉语古典文本资料库[12]有 13000 种文本，10 万卷，近 13 亿字，大小为 3.14 GB, 基本上涵盖了所有朝代的古籍文献，本文选定这个数据集作为筛选古汉语字词的依据。为了达到良好的搜索性能，首先在 Apache Elasticsearch 7.4.2 中将 3.14 GB 的古汉语语料库建立索引，然后通过 Elasticsearch 提供的检索 API 对词典中的字和词进行筛选。如果某个字或者词语能在 Elasticsearch 索引库中检索到，就将这个词标记为文言词汇，最终形成一个只有古汉语字词的候选词表。

对于 Elasticsearch 中索引和检索所用的分词组件，本文使用了 IK 分析插件[31]，将上文得到的字和词转换成字典文件，并替换 IK 中原有的字典文件，analyzer 配置为 IK 的优点是能够精确地按照中文词典分词，Mapping 设置如下，

```
{
  ″properties″: {
    ″content″: {
       ″type″: ″text″,
       ″analyzer″: ″ik_smart″,
       ″search_analyzer″: ″ik_smart″
         }
     }
}
```

Elasticsearch 有三种搜索方式，term，match 和 match_phrase，其中 match 和 match_phrase 都会对搜索关键词进行拆分，然后对子关键词再进行检索，这样就无法满足本实验的要求，而 term 搜索方式能够做到整词匹配，虽然 term 搜素方式会将字典中不存在的词语进行单字拆分，但是这不影响本实验的结果，因为本实验针对的是字典中的词进行检测，没有字典之外的词语。接下来经过去重处理和繁体字转换，得到了一个包含 331516 个字词的词典，本文称之为基础词典，其中词语有 66712 个。由于字对分词的准确度的影响并不高，所以本文重点统计了基础词典中的两个字以上的词语的一些分布特征，横坐标代表词语在 3.14 GB 语料中出现的次数，纵坐标代表这个频率的词有多少个，如图 2 所示。

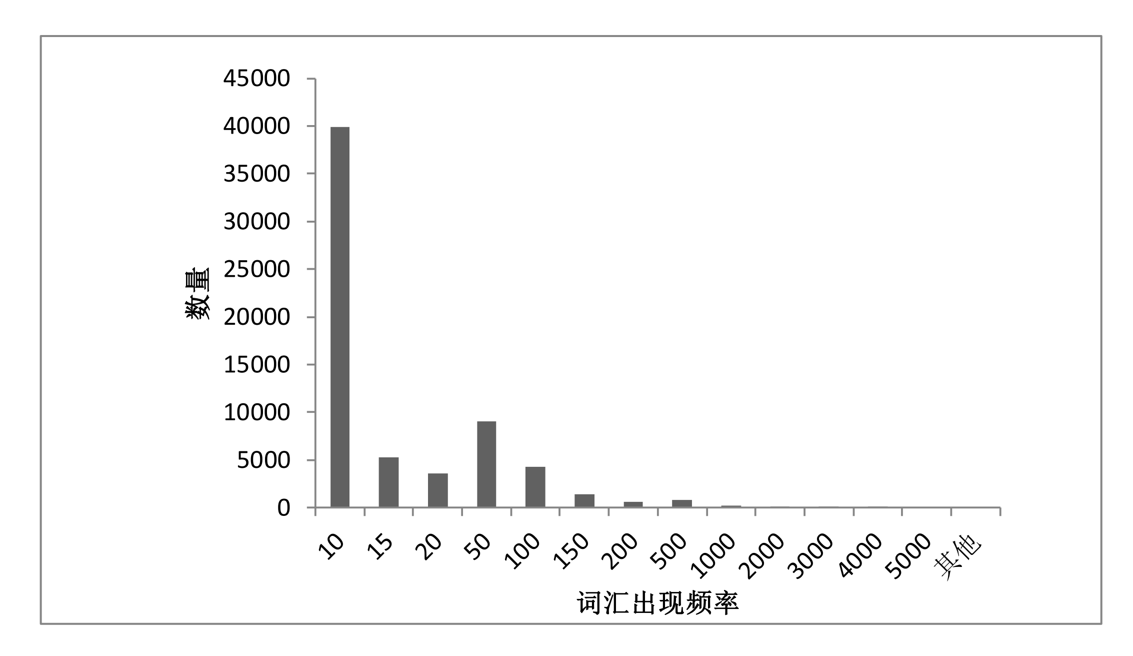


图 2 基础词典不同词频的词分布

由上图可以看出，频率在 10 左右的词汇占了大约 4 万个，词频在 10 以上的词汇的数量比较少。

2.2 新词发现

新词发现是自然语言处理的重要的一项技术，用于抽取字典中没有的新词[13]。在获取古文基础词典后，仍然难以确保基础词典收录了全面的古汉语字词。为了弥补基础词典的不足，在这一步骤中，本文提出了多特征融合的方法实现新词发现，综合采用 N-Gram 词频、互信息、信息熵、位置成词概率相结合的方式在大规模语料库中抽取新词，语料库采用上文提到的开源的 3.14GB 的汉语古典文本资料库。在既往演剧中也有采用新词发现的方式抽取古汉语词汇 [5]，但是这些方法均在少量的语料库上进行实验，效果并不理想，而在本实验中采用大规模语料，并使用分布式计算平台 Apache Spark 进行新词发现。

N-Gram 是一个统计语言模型[14]，它的假设是，第 N 个词的出现只与前面 N-1 个词相关，而与其它的词都不相关。本文首先对古汉语文本进行切分，切分后的每个单元被认为是一个 gram,进而统计出每个 gram 的频率。根据设定的阈值，过滤掉不符合词长要求的词语。使用 N-Gram 的优点是语言无关性，不需要对古汉语文本进行语言学处理。

互信息和信息熵都是信息论中的概念[15]，互信息可以计算两个对象的关联程度，如果 X 和 Y 互相独立，那么 X 和 Y 之间互相不提供任何信息，他们的互信息就为 0。N-Gram 获取高频率的文本片段，但是一个文本片段出现的频率高并不能代表这个文本片段是一个真正的词语，它可能是多个词语结合在一起的词组。本文采用互信息度量不同文本片段的凝固程度。互信息的计算公式如下，

$$MI(w_1 \cdots w_n) = \log_2 \frac{p(w_1 \cdots w_n)}{avg(w_1 \cdots w_n)} \quad (1)$$

$$p(w_1 \cdots w_n) = \frac{f(w_1 \cdots w_n)}{num} \qquad (2)$$

$$\text{avg}(w_1 \cdots w_n) = \frac{1}{n-1} \sum_{i=1}^{n-1} p(w_1 \cdots w_i) p(w_i \cdots w_n) \quad (3)$$

其中，$w_1 \cdots w_n$代表由多个字构成的词语，$p(w_1 \cdots w_n)$代表词语$w_1 \cdots w_n$在语料库中出现的概率，$f(w_1 \cdots w_n)$ 代表词语$w_1 \cdots w_n$在语料中出现次数，num 表示语料库的字数，avg(w1 wn) 表示词语中的字不同组合的平均概率。

信息熵也被称为自由度，用于判断古汉语文本左右相邻字符的相互关系的稳定性，熵越大，稳定性越小，越有可能称为独立的词。信息熵分为左信息熵和右信息熵。左信息熵代表一个文本片段与左边的字符相结合的稳定程度，右信息熵代表一个文本片段与右边的字符相结合的稳定程度，计算公式如下，

$$H_{left}(W) = -\sum_{w_{left} c_{left}} p(w_{left}|w) \log_2 p(w_{lefl}|w) \quad (4)$$

其中，$H_{left}(W)$是候选词语w左信息熵，c_{left}是候选词w 左边相邻的字符集合，$p(w_{left}|w)$是出现候选词w时，其左边相邻字符w_{left}的条件概率。$p(w_{\text{left}}|w)$的计算公式见式（5），

$$p(w_{\text{left}}|w) = \frac{N(w_{\text{left}})}{N(w)} \qquad (5)$$

其中，$N(w_{\text{left}})$是左边相邻字符w_{left}及候选词w共同出现的概率，$N(w)$是候选词w单独出现的概率，同理，右信息熵可判断词语右边界，候选词右信息熵计算方法如公式（6）所示，

$$H_{right}(W) = -\sum_{w_{right} c_{right}} p(w_{right}|w) \log_2 p(w_{right}|w) \quad (6)$$

其中，$H_{right}(W)$是候选词语w右信息熵，c_{right}是候选词w*右边相邻的字符集合*，$p(w_{right}|w)$是出现候选词w时，其右边相邻字符w_{right}的条件概率。$p(w_{\mathrm{right}}|w)$的计算公式见式（7），

$$p(w_{\mathrm{right}}|w)=\frac{N(w_{\mathrm{right}})}{N(w)} \qquad (7)$$

其中，$N(w_{\mathrm{right}})$是右边相邻字符w_{right}及候选词w共同出现的概率，$N(w)$是候选词w单独出现的概率。

由于新词发现需要将在整个语料库加载到内存上进行计算，只有在整个语料库上重复出现过的字符串才可能是候选词，因此语料库越大，所需要的内存也越多，时间复杂度也越大。本文基于 Spark 实现了并行处理[16]，在效率上有了很大提升，算法基本流程如表 1 所示，

表 1 新词发现算法基本流程

算法基本流程
输入：原始语料 D＝{S 1，S 2，…，S n} 输出：候选词的集合 ① 将原始语料文件以RDD变量的形式读入内存，假设该RDD变量为corpus_rdd。 ② 对corpus_rdd的每一行，把出现的每一个字符存到数组中，并将每个字符映射到频率（也就是 1 ）。最后用Reduce操作把相同字符的频率相加，得到每个元素是（字符，频率）的RDD量nGram_RDD。 ③ 对RDD变量nGram_RDD做filter操作，将频率大于阈值的词保留，得到新的RDD变量nGram_filter_RDD。 ④使用map操作对RDD变量nGram_filter_RDD中的每个词计算信息熵。 ⑤ 使用map操作计算好词频和信息熵的RDD，建立Trie树。 ⑥ 使用broadcast操作将建好的Trie树广播到其他节点，减少计算中的通信开销。 ⑦ 使用map操作计算互信息，并过滤掉低于互信息和信息熵阈值的词语，返回最终的字典集合。

Trie 树是一个树形的结构，也是哈希树的变种[23]。Trie 树利用前缀来缩短检索时间，能够最大限度地减少字符串的比较，尤其在动态地增加或者修改数据的场景下性能表现更好。本文通过 Spark RDD 实现了分布式的 Trie 树，在效率上有了提升[17]。

RDD 是弹性分布式数据集[18]，是 Spark 实现分布式处理的基石，RDD 能将数据分布在多台机器上，提高了计算性能。本文采用的 Spark 集群采用 1 个 master 主控节点，3 个 slave 从节点组成，节点配置如表 2 所示，

表 2 节点配置

项目名词	说明
CPU	8 Core, Interl Xeon
Memory	64GB
Disk	1TB
OS	Ubuntu 18.04.3 LTS
Java Version	JDK 1.8
Hadoop Version	3.2.0
Spark Version	2.4.4

在算法中涉及了算法的超参数配置，本文选取的参数阈值如表 3 所示，

表 3 参数配置

参数	参数值
最小词频	10
最大词长	2
最小词长	8
互信息阈值	0.2
信息熵阈值	0.2

通过运算，以表 4 所示的样式输出，

表 4 算法输出结果

信息熵	互信息	词频	词语
5.125501	200.1129	61	龃龉
4.775116	97.83365	104	涅槃
4.569175	167.3033	63	徘徊
3.947703	653	18	邂逅
4.36353	240.5139	41	邯郸
3.459432	1127.909	11	阛阓
3.563074	477.1923	25	琵琶
3.721612	589.3325	19	窀穸
3.25	775.4375	16	袈裟
4.682079	194.9931	43	匍匐
3.560935	275.7111	38	鹧鸪
4.026244	180.8911	51	糟粕
3.640224	653	15	鹦鹉
2.867634	563.9545	22	婕妤
3.418296	827.1333	12	澳漻
3.558519	438.6313	21	玲珑
3.741446	437.6367	20	鸳鸯
3.471354	496.28	19	盩厔
3.886842	347.049	24	囹圄

至此，通过词频统计并搭配互信息，信息熵得到的词典已经涵盖了足够多的候补词语，但是仍然存在一些非古汉语词语，这些词语有些是完整词语的部分片段，例如，“氏家世”并不是一个完整的词语，一般用法为“刘氏家世”，“家世”，为了进一步提高成词的准确度，本文通过位置成词概率[7]对上一步得到的词典进行过滤。在古汉语中，每个字或者词语都有自己的构词规律，某个字会出现在合成词的固定的位置，例如“才”一般出现在某个词的结尾，“秀才”“王进才”，所以在结尾这个位置的概率比较高。位置成词概率公式为，

$$\mathrm{PWP}(w, \mathrm{pos}) = \frac{N(w, \mathrm{pos})}{N(w)} \qquad (8)$$

其中，pos 表示古汉字 w 在该词中出现的位置，位置包括词首，词中，词尾，$N(w, \mathrm{pos})$ 表示 w 出现在词语中的位置 pos 的所有词语的频次；$N(w, \mathrm{pos})$则表示 w 在基础词典的词语中出现的总次数，根据这个公式，本文基于上文得到的基础词典计算出位置成词概率表，如表 5 所示，

表 5 位置成词概率表

古汉字	词首概率	词中概率	词尾概率
耀	0.234848	0.037879	0.727273
老	0.547358	0.188435	0.264207
考	0.494881	0.051195	0.453925
耄	0.526316	0	0.473684
者	0.052805	0.29703	0.650165
耆	0.694118	0.070588	0.235294
耇	0	0	1
耈	0	0	1
耋	0.411765	0	0.588235
而	0.031208	0.913161	0.055631
耍	0.642857	0.042857	0.314286
耎	0.416667	0	0.583333
耏	0.5	0	0.5
耐	0.606061	0.090909	0.30303
耒	0.333333	0	0.666667

假设一个词语词首含有某个古汉字，这个古汉字在词首的位置的概率越小，表明这个词语成为古汉语词语的可能性越小，同理，如果一个古汉字在词尾的位置的概率越小，表明这个词语成为古汉语词语的可能性也越小。

最终，本文在基于互信息和信息熵得到的词典中通过计算首字和尾字的位置概率，将不合理的词再次过滤，得到最终的词典，本文称之为候补词典。

在候补词典中，本文统计了不同长度的词语的数量分布，如图 3 所示，

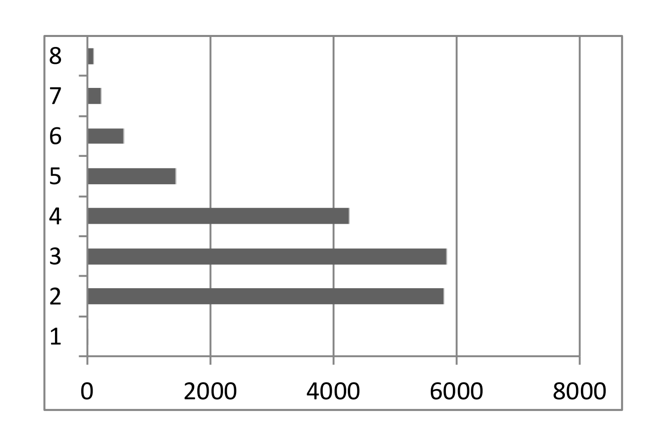


图 3 候补词典中不同长度的词语的数量比较

由此可见，在候补词典中，二字词和三字词占比最多，并且二字词和三字词数量接近。

2.3 分词器的构建

中文的分词算法[19]有很多，根据它们的原理，可以分为三种，基于词典的分词，基于统计的分词，基于理解的分词，其中基于词典的分词的算法的分词速度是最快的。基于词典的分词算法主要有正向最大匹配，逆向最大匹配，双向最大匹配。

甲言是 github 上开源的古汉语分词器，是一个完整的分词系统，主要分词方式有两种，一种是正向最大匹配分词，另一种是使用训练好的 HMM[4]模型来分词。

在获取基础词典和候补词典后，我们将两个词典整合在一起，其中有一部分会有相同的词语，这一部分词语通过冗余处理合并，最终形成词典并作为分词器的分词词典。本文使用正向最大匹配作为分词算法[24]。

3 结果与分析

3.1 测试数据集

为了测试古文分词的准确性，就需要一个已标注好的语料库作为测试依据。宾夕法尼亚州语言数据联盟大学网站[20]提供了人工标注的文言语料库(LDC2017T14)，该语料库是对左传的分词和词性的标注，共包含了 180,000 个中文字符，由两部分构成，训练数据（166138 字）和测试数据（28131 字），数据格式为纯文本文件，如表 6 所示。

表 6 左传语料库样例

左传语料库样例
元年/t ，/w 春/n ，/w 王/n 正月/t 辛巳/t ，/w 晉/ns 魏舒/nr 合/v 諸侯/n 之/u 大夫/n 于/p 狄泉/ns ，/w 將/d 以/c 城/n 成周/ns 。/w 魏子/nr 蒞政/v 。/w 衛/ns 彪傒/nr 曰/v ：/w “/w 將/d 建/v 天子/n ，/w 而/c 易位/v 以/c 令/v ，/w 非/d 義/v 也/y 。/w 大事/n 奸/v 義/n ，/w 必/d 有/v 大咎/n 。/w 晉/ns 不/d 失/v 諸侯/n ，/w 魏子/nr 其/u 不免/v 乎/y ！/w ”/w

然而该数据只包含左传的数据，数据的多样性略显不足。我们扩展了一些数据集，首先选取一些代表性的，不同朝代的文言语料，其中包括春秋，战国，秦朝，汉朝，魏晋南北朝，隋唐，宋朝，辽金元，明朝，清朝，然后组织了古汉语专业学生进行人工标注，最终与左传的数据集进行合并，作为测试集。选取的古籍文本出处如表 7 所示，

表 7 标注的文本的出处

朝代	节选出处
春秋	《史记・田敬仲完世家》《史记・晋文称霸》《论语・学而篇》《邓析子・无厚》《史记・项羽本纪》
战国	《史记・孟尝君列传》《全上古・上书说秦昭王》《楚辞・离骚》《孟子・梁惠王、公孙丑》》
秦朝	《史记・秦始皇本纪》《韩非子・存韩・上书韩王》《史记・李斯传・上书对二世》
汉朝	《论衡》《汉书・董仲舒传》《贾谊传》《史记・仓公传》
魏晋南北朝	《傅子》《抱朴子》《三国志・魏书・武文世王公传》《陆景》《典语》
隋唐	《史通・自叙》《长乐老自叙》《与文徵明书》《先侍御史府君神道表》
宋朝	《金石录后序》《指南录后序》《训俭示康》《九议》
辽金元	《辽史》《归潜堂记》《金史》《元史・本纪第一》

明朝	《御制皇陵碑》《前历试卷自序》《白牛生传》《立命之学》
清朝	《三十自述》《三依赘人广自序》《弢园老民自传》《与弟文韶书》

标注方法是将原始文本中的字，词，标点符号使用空格分开，标注样例如表 8 所示，

表 8 人工标注样例

人工标注样例
元年王充 者 ， 会稽 上虞人 也，字 仲任 。 其先 本 魏郡 元城 一姓 。 孙一几世 尝 从军 有功 ， 封 会稽 阳亭 。 一岁 仓卒 国绝 ， 因 家 焉 。 以 农桑 为业 。 世祖 勇 任气 ，卒咸 不揆 於 人 。 岁凶 ，横道 伤杀 ， 怨仇众多 。 会 世 扰乱 ， 恐 为 怨仇 所擒 ， 祖父 泛 举家 檐载 ， 就 安 会稽 ， 留 钱唐县 ， 以 贾贩 为 事 。 生子 二人 ，长 曰 蒙 ，少 曰 诵 ， 诵即 充父 。 祖 世 任气 ， 至 蒙 、 诵 滋甚 。 故 蒙 、 诵 在 钱唐

由于左传的标注格式区分了词性，所以这里还需要将左传的标注格式转换成空格标注的格式，如表 9 所示，

表 9 左传语料库标注格式转换后的格式

左传语料库标注格式转换后的格式
元年 ， 春 ， 王 正月 辛巳 ， 晋 魏舒 合 诸侯 之 大夫 于 狄泉 ， 将 以 城成周 。 魏子 莅政 。 卫 彪傒 曰 ： “ 将 建 天子 ， 而 易位 以 令 ， 非 义 也 。 大事 奸 义 ，必 有 大咎 。 晋 不 失 诸侯 ， 魏子 其 不免 乎 ！ ”

3.2 词典扩展前后的效果比较

由于古汉语和现代汉语在验证准确度的方法上是一样的，所以本文使用了 Bakeoff 2005 数据集[21]包含的 perl 脚本，Bakeoff 是国际计算语言学会（ACL）中文语言处理小组 SIGHAN 所主办的国际中文语言处理竞赛，而 Bakeoff 2005 是 SIGHAN 于 2005 年在韩国

济州岛举行的第二届竞赛的数据集，该数据集是完全免费和公开的，但是使用的前提是非商业使用。在该数据集中 scripts 文件夹下包含了一个用于测评的脚本文件 score,该文件是一个 Perl 程序，为了能让程序运行，需要有一个 Perl 编译器，而 Ubuntu 18.04.3 LTS 自带了 Perl 编译器和运行环境，所以本文将在 ubuntu 下进行分词的评估。该脚本需要输入三个参数，分别为标准词表，标准切分，待评测的切分[22]。

本文采用准确率（Precision）、召回率（Recall）和 F 值（F-measure）作为分词的评价指标，其计算公式为：

$$P = \frac{N \cap M}{N} \times 100\% \quad (9)$$

$$R = \frac{N \cap M}{M} \times 100\% \quad (10)$$

$$\mathrm{F} = \frac{2\mathrm{PR}}{P + R} \times 100\% \quad (11)$$

其中，N 表示实验获得的新词总个数，M 表示语料中存在的新词总个数

接下来，本文基于 Bakeoff 提供的脚本分别对基础词典，候补词典，合并词典进行评估。评价过程如表 10 所示，

表 10 评价过程

评价过程
输入： 待切分的语料库ancient_original_data.txt，标准切分文件ancient_gold_data.txt，基础词典ancient_basic_dict.txt，候补词典ancient_addtion_dict.txt，合并词典ancient_dict.txt 输出： 评价指标，准确率（Precision）、召回率（Recall）和F 值（F-measure）

①准备待切分的语料ancient_original_data.txt，该语料由上文提到的测试数据集转换而来，将测试数据集中的分词标记删除得到原始语料。

②使用上文编写的Python分词器对待切分语料进行分词，分词器使用基础词典ancient_basic_dict.txt，分词算法使用正向最大匹配。

③输出已分词的古汉语文本ancient_segmentation_data.txt

④将已分词的古汉语文本ancient_segmentation_data.txt，标准切分文本ancient_gold_data.txt，标准词典ancient_dict.txt输入Perl评测脚本并在Ubuntu上运行脚本。

⑤ 输出使用基础词典分词的评价指标，准确率、召回率和F值。

⑥ 从②重新开始，并使用候补词典ancient_addition_dict.txt，得到候补词典分词的评价指标。

⑦从②重新开始，并使用合并词典ancient__dict.txt，得到合并词典分词的评价指标。

在使用正向最大匹配分词算法的条件下，基础词典，候补词典和合并后的词典的分词结果比较如表 11 所示，

表 11 三个词典的分词结果比较

指标	基础词典	候补词典	合并词典
P	0.801	0.630	0.821
R	0.838	0.778	0.859
F	0.819	0.696	0.839

从上面的比较可以看出，将基础词典和候补词典合并之后，分词器的准确性得到了很大的提升。

3.3 与古汉语开源分词器比较

甲言是一个开源的古汉语分词器，为了验证本文提出的分词器是否能够达到开源分词器的分词效果，本实验将本文的古汉语分词器与甲言分词器进行比较，甲言分词器中包含了两种分词算法，一种是基于词典的分词，一种是基于 HMM 模型的分词。因此，我们分别

与这两种方式进行比较。评价方式与上文提到的评价过程类似，将待切分的文本ancient_original_data.txt 分别输入两个分词器，然后分词，并输出各自的评价指标，准确率、召回率和 F 值，结果如表 12 所示，

表 12 三个词典的分词结果比较

分词器	P	R	F
本文的分词器	0.821	0.859	0.839
甲言的词典分词	0.651	0.751	0.698
甲言的 HMM 分词	0.750	0.798	0.773

由此可以看出，本文设计的分词器与甲言的两种分词模式相比，在准确度，召回率，F 值这些指标上均有明显地提升，本文提出的分词器的 F 值比甲言的词典分词模式高出了 14%，总体上取得了良好的效果。由此表明，基于互联网大规模语料库的词典构建在古汉语分词上是可行的。

4. 总结

本文利用大规模古文语料库，通过整合了互联网上的古汉语数据资源，生成了一个古汉语分词基础词典，并使用 N-Gram、互信息、信息熵、位置成词概率相结合的新词发现的方式在大规模语料库上抽取古汉语词汇，弥补了基础词典的不足。在词典的基础上，本文利用正向最大匹配实现古文的分词，通过与甲言进行比较，在人工标注的分词语料库上取得了良好的效果。

本研究也存在一定的局限。例如本文采集古汉语数据所用的数据源比较少，在以后的研究中可以进一步扩大数据源，收集更全面的古汉语数据；在新词发现中的超参数的设置还有待优化，以致于进一步提升分词的准确度；在本文中没有考虑歧义词处理，然而在歧义词处理方面，刘风成[30]提出 AdaBoost.MH 算法进行语义消歧，并引入了语义范畴,提高了排歧的正确率。

中国古代文献是一个宝藏，从古汉语文本中挖掘有价值的信息在考古中有很重要的意义。本文利用在线大规模古文语料库，经过数据处理，构建古汉语词典，利用正向最大匹配实现对古文的分词。通过与甲言分词系统进行比较，分词准确率有了较大的提高。通过

对古文献的分析研究，有助于深入了解中国文化的变迁，开展数字心理考古研究，为进一步弘扬传统文化提供技术支撑。

参考文献：

[1] Amrani, A., V. Abajian, Y. Kodratoff, and O. Matte-Tailliez. "A Chain of Text-Mining to Extract Information in Archaeology." 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 7-11 April 2008 2008.

[2] 严顺. 基于 CRF 的古汉语分词标注模型研究[J]. 江苏科技信息, 2016(8):10-12.

[3] 王晓玉, 李斌. 基于 CRFs 和词典信息的中古汉语自动分词[J]. 数据分析与知识发现, 2017, 1(5).

[4] 钱智勇, 周建忠, 童国平, et al. 基于 HMM 的楚辞自动分词标注研究[J]. 图书情报工作, 2014(04):107-112.

[5] 李筱瑜. 基于新词发现与词典信息的古籍文本分词研究[J]. 软件导刊, 2019, 18(04):66-69.

[6] 刘永楠, 李建中, 高宏. 海量不完整数据的核心数据选择问题的研究[J]. 计算机学报, 2018.

[7] 林自芳, 蒋秀凤. 基于改进位置成词概率的新词识别[J]. 福州大学学报(自然科学版), 2011(01):48-53.

[8] 夭荣朋, 许国艳, 宋健. 基于改进互信息和邻接熵的微博新词发现方法[J]. 计算机应用, 2016, 36(10):2772-2776.

[9] 古汉语分词器，甲言，https://github.com/jiaeyan/Jiayan

[10] Prabhu C., Chivukula A., Mogadala A., Ghosh R., Livingston L. (2019) Big Data Tools—Hadoop Ecosystem, Spark and NoSQL Databases. In: Big Data Analytics: Systems, Algorithms, Applications. Springer, Singapore

[11] Olsson, J. (2019). Using Elasticsearch for full-text searches on unstructured data (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-395654

[12] 汉语古典文本数据库 https://github.com/mahavivo/scripta-sinica

[13] 王思丽,祝忠明,刘巍,杨恒.领域本体学习语料的自动获取与预处理方法研究[J].图书馆学研究,2019(20):54-64.

[14] H. Wang, B. Wang, M. Zou and J. Duan, "New Cyber Word Discovery Using Chinese Word Segmentation," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 970-975.

[15] Cheng, & Lee, W. (2018). Combining Mutual Information and Entropy for Unknown Word Extraction from Multilingual Code-Switching Sentences.

[16] Junjie Hou, Yongxin Zhu, Sen Du, Shijin Song, Design and implementation of reconfigurable acceleration for in-memory distributed big data computing,Future Generation Computer Systems.

[17] 刘鹏, 滕家雨, 丁恩杰, et al. 基于 Spark 的大规模文本 k-means 并行聚类算法[J]. 中文信息学报, 2017(04):150-158.

[18] Aziz K., Zaidouni D., Bellafkih M. (2018) Big Data Optimisation Among RDDs Persistence in Apache Spark. In: Tabii Y., Lazaar M., Al Achhab M., Enneya N. (eds) Big Data, Cloud and Applications. BDCA 2018. Communications in Computer and Information Science, vol 872. Springer, Cham

[19] 张梅山, 邓知龙, 车万翔, et al. 统计与词典相结合的领域自适应中文分词[J]. 中文信息学报, 2012(2):8-12.

[20] 古汉语语料库，https://catalog.ldc.upenn.edu/LDC2017T14

[21] 中文分词大赛数据集，http://sighan.cs.uchicago.edu/bakeoff2005/

[22] 黄翼彪. 开源中文分词器的比较研究[D]. 郑州大学, 2013.

[23] 王思力, 张华平, 王斌. 双数组 Trie 树算法优化及其应用研究[J]. 中文信息学报, 2006, 20(5):26-32.

[24] 熊志斌, 朱剑锋. 基于改进 Trie 树结构的正向最大匹配算法[J]. 计算机应用与软件, 2014(05):282-284.

[25] 汉文学网，https://www.hwxnet.com/

[26] 汉典网，https://www.cidianwang.com/

[27] 汉典，https://www.zdic.net/
[28] 在线汉语字典，http://xh.5156edu.com/
[29] 国学大师，http://www.guoxuedashi.com/
[30] 刘风成, 黄德根, 姜鹏. 基于 AdaBoost.MH 算法的汉语多义词消歧[J]. 中文信息学报, 2006(03):8-15.
[31] IK 插件，https://github.com/medcl/elasticsearch-analysis-ik